by Joe Cooper, carm_popcomm@hotmail.com

You Can Do It—Computer Programming Made Easy

As I have been outlining in this column for the past several months, radio receiving and transmitting technology has undergone a significant transformation since the beginning of this decade. That transformation has been the switch from the use of "real" electronic components to "virtual" components in a new type of radio design called Software-Defined Radio (or SDR).

SDR works by sampling large bandwidths of radio signals directly off the antenna, turning that information into digital form and then processing that digital information through digital signal processing (or DSP) software in a computer.

As in the case of any new technology, the first casualty is generally the dominant technology that is replaced. Currently the primary radio monitoring technology that will be the most directly affected will be those radios using super-heterodyne (or simply, superhet) circuitry, which have been mass produced since the 1920s.

The nice thing about the old superhet radios was that they were simple to build, so anyone with a good understanding of electronic circuits could put one together. Even for those who found the superhet radio complicated, there were even simpler designs available, such as crystal radios. Therefore, the true foundation for today's radio monitoring hobby is that just about anyone can put together a working radio.

If you look through one of the many books or magazines about simple home-built radio designs published during the early to mid-20th Century, you'll see hundreds of plans and projects for crystal radios, tuned-radio frequency (TRF), or regenerative radios. Radio monitoring in the 20th century was a real reflection of democracy because after 1920 everyone in the United States had unrestricted access to radio parts, plans, and expertise. That democratic ideal was displayed during periods of wartime when citizens "of the free world" were allowed to keep their shortwave monitoring radios and could listen in to "real-time" radio activity taking place on the battle fields of Europe, North Africa, and the Pacific.

Nor were there restrictions placed on listening in to propaganda broadcasts, first from the Axis powers during the 1930s and '40s and later from Communist countries during the Cold War. Contrast that with the experiences of people living in Germany, Italy, or Japan during the height of Fascist power, or in the Soviet Union during the Stalinist period. During those times, listening to "unauthorized" frequencies was punishable by brutal treatment, imprisonment, and even death.

The "Kit Radio" Legacy Of Freedom

So all those Heathkit and Knight kit radios that were sold during the 1950s, '60s, and '70s were not just an inexpensive and easy way for people to become involved in the radio monitoring hobby, those simple-to-build radios were a tangible symbol of the democratic freedom that so many people had fought and died for during World War II. Their manufacture and distribution showed the difference between our political system and those of communist and totalitarian nations.

While today's radio technology, with its emphasis on integrated circuits and miniaturized components, has made construction slightly more complicated, there are still several companies selling radio kits to the public. However, the truth is that digital technology dominates the fields of radio and electronics today; not just for receiving, but for transmitting as well. Like it or not, time is running out for both the AM and FM modes because new digital modes like DAB (Digital Audio Broadcasting), DRM (Digital Radio Mondiale), and HDC (Hi-Definition Compression) are going to be the new modes of radio transmission.

Working With Today's Technology

There is, however, a problem with today's SDR receiver and digital modes of radio transmission. It's that the former democratic accessibility the old radio technology provided is going to change radically—and very soon. This is because digital signals must be processed using computer software rather than through electronic components, and not everyone can work with digital technology as easily as they did with simple technology like crystal radios.

There will still be hardware components in SDR technology, so people will still need their soldering guns to put together *part* of an SD radio. However, these parts will be used only to sample radio signals and then ship the resulting digital information off to a computer via a USB port. But from now on, the real "innards" of any SDR will always be a personal computer, an SDR software package, and a computer sound card. The question this raises is whether this change will also bring about a serious curtailment of hobbyist radio construction and tinkering because of the cost and complexity of the tasks involved.

Certainly in the case of hardware tinkering the number of opportunities available for "soldering iron jockeys" will decrease dramatically, particularly when digital modes of transmission become the norm in 10 or 20 years (if not sooner). However, that does not mean that certain forms of "hands-on" tinkering will not be possible.

While SDR technology may be closing a chapter of radio history, it is also opening up a new one. This will be a new hobbyist field of software programming for SD radios.

The Challenge Of Computer Programming

I know that many of you out there may be saying "who, me—a software programmer?" and the answer is most definitely "yes, you!" Frankly, not only am I saying this, but so are many other people within the computer and radio industries.

Like it or not, computer programming is going to become the new literacy and will determine how you participate in the world of the 21st century. I know a lot of people were unable to

get their VCRs to stop flashing "12:00" back in the 1980s and '90s, and as with any new technology there will be people who will not be able to adjust to this. Still, most of you, if you actually sit down and invest a little time to read and experiment, will find (much to your surprise) that you can become a fairly decent computer programmer in less time than you may expect.

The main reason I make this claim is because you now have at your fingertips more support, information, and resources—and for free—than at any other time since the advent of the personal computer. Plus, the Internet has changed how people obtain information and the extent and range of information that's available. And more importantly, there are many people out there who want you to succeed, and that includes the leaders of the computer industry such as Bill Gates of Microsoft.

In the upcoming series of columns I'll outline how you can set yourself up to succeed in learning how to be a good computer programmer using free or inexpensive training resources. And, since you're reading this magazine, you already have a tremendous advantage over most people when it comes to learning how to program computers. Specifically, that advantage is that you can read English! When it comes to computer programming, the truth is that the vast majority of the *good* training material on the market today is *only available* in English.

Let's start at the beginning—what do you need in order to become a good computer programmer?

How To Become A "Hobbyist" Computer Programmer

Some of you may be surprised that, in my experience, the most important thing you need to become a good computer programmer (as opposed to someone who is simply a software hacker) is a well-defined goal for your programming tasks.

Unless you really want to go into a career as a computer software designer or commercial application developer, you really do not need to know everything that there is to know about programming. Nor do you need to know every single programming language available or every technique or process used to put all the bells and whistles common in today's computer software.

All you need to do is focus on one particular area that will give you the most reward for the least amount of time and effort spent on your computer. Remember, the real purpose of this entire exercise is to give you better access to, and control over, your SD radio. Therefore, you should first establish the make, model, or type of SD radio you want to be working with and start building your skills around that particular radio. It does not even have to be as sophisticated as an SD radio, because there are many computer-controlled radios out there that can be the foundation for your programming exercises.

As I've outlined in past columns, computer-controlled radios are relatively simple to program since all you are doing is placing a particular numerical value into the radio's built-in computer via a serial cable. These numerical values then trigger a particular function, such as changing frequencies, adjusting bandwidth, or changing the mode (AM, FM, SSB or CW) of transmission that is being monitored. In most cases the programming requirements for a computer-controlled radio are very simple and can be satisfied by a relatively easy software language, such as BASIC. And the novice computer programmer will appreciate that there are several versions of BASIC available for free through Internet downloads (see the sidebar accompanying this column).

First develop a simple strategy in your approach to becoming a programmer. The simplest and most straightforward approach is to make your point of focus the particular SD radio you happen to own.

What You Need To Know

Even for those only interested in radio monitoring (rather than, say, hams who might also want to transmit with SDR technology), the number of receive-only SD radios is still very limited. Also, some of those SD radios are primarily designed to work with the LINIX operating system, which while stable and popular, is really best tackled by someone with a more advanced background in computing.

In this column I'm going to be focusing on the Flex Radio System's SDR-1000R since the company actively supports the concept of open source software (OSS), meaning the programming code used to create a software application is made freely available. OSS allows anyone with programming experience to revise or change the programming code so they can modify the code to build new versions of customized software. This is valuable because many people can bring their unique skills to help develop a software program in ways that may not occur in traditional software development settings, where cost and "time-to-delivery" limitations normally restrict the number of development cycles.

Moreover, the unique nature of SDR technology allows for the *independent* development of software programs because they are hardware independent. In fact, you may have three programmers come up with three very different software packages that work equally well with the same hardware component. This is where the ability to experiment and customize SDR software becomes an important motivation for getting involved.

Building Your Knowledge Toolbox

It is important to understand where you're going to get your new "toolbox" to begin building "virtual" radios. There are many tools that will help you become an amateur computer programmer; some are tangible (program editors, compilers, and debuggers) and some intangible (knowledge, skill, and experience). Just remember that the acquisition of programming skills does not come overnight, but rather through a series of steps, which if systematically perused, will deliver good results over time, much like learning a musical instrument. And just as there are many forms of music, such as jazz, rock, classical, and folk, there are many forms of computer programming languages, such as Visual Basic, C+, Java, and assembly language.

It is beyond the scope of this column to go into detail about each of these languages, or about how to create a computer program. I will, however, give you some pointers on where to find more information. I'll also point you to a number of free, or at least reasonable, resources. You don't need to spend a fortune on books or programming software that can become obsolete very quickly, simply to get started.

I am assuming as you lay out an entry-level strategy for learning computer software programming that it will be with the aim of becoming a hobbyist level programmer. So before you do anything, you need an idea of what computer software programming is, and is not. This will give you a good understanding of the "culture" of computer programming, which has its own language, logic, and rules, and you'll

need a basis in it to be able to read (and appreciate) your next level of training.

In the sidebar you will find a series of suggested titles that are currently available through many bookstores or your local public library. These are just a few of the slew of books (not to mention training videos, audio CDs and DVDs) that are available. One that I would strongly suggest is *Code: The Hidden Language of Computer Hardware And Software* by Charles Petzold. While the text is a bit out of date, his overview of how computer programming has developed is still very worthwhile reading as he talks about computer technology and its direct link to the human need to communicate. This is a point that you need to understand because computers are more than just mechanical devices that make calculations; they are tools of communication that are transforming the culture and civilization of the entire world.

As I have pointed out in this column, today's common microcomputer developed out of a complex set of social and economic events, including accidents, ironic circumstances, and just plain human luck. As Petzold points out, microcomputer technology has not developed out of a rational plan in a laboratory setting, but through the work of people who find a technology and discover hidden potentials within it (hence the title of his book). More importantly, Petzold, who holds solid programming credentials, takes the reader through a detailed (but understandable) explanation of how software code actually "works" inside the computer, performing various operations within its CPU and other components. Truly, if you want a proper understanding as to why you are writing lines of computer code when you build a software program, this is the place to start.

After you've become familiar with the general theory and practice of computer programming, it's time to begin learning how to build a software program. Again, there are many books that will provide you with an understanding of the different parts of a programming language before you go on to learn a specific programming language. One I'd like to bring to your attention to is *Learning Computer Programming* by Mary E. Farrell. The subtitle for this book is interesting: "It's not about languages." In saying that Farrell is raising a very important point that anyone who is starting to learn computer programming should remember. It's not the language that makes a computer program do what it does, but rather the sequence of concepts and the logic behind them that the programmer uses, through whichever programming languages that best serves his or her needs.

When trying to learn computer programming, people often find what they encounter is arcane and downright frustrating. But by looking at the "why" of computer programming, rather than the "how" of languages, Farrell provides you with the "thinking tools" that you will need to undertake real software programming. Only after you've mastered the internal processes common to all modern programming languages does Farrell introduce you to how specific programming languages (C++ and JAVA) are used and how they differ in their approach.

A Guide To Hobbyist Level Computer Programming

Many people who become interested in learning how to program a personal computer on a casual basis find that a lot of the training books and material available are too complicated to really help them get started. The following books and materials are geared toward those people who simply want to understand computer programming at a beginner level and maybe create simple software programs for their own use. Check a few of them out.

General Introduction To Computer Programming

You Can Do It!: A Beginner's Introduction To Computer Programming, Francis Glasborow, John Wiley and Sons, 2003.

Sams Teach Yourself Beginning Programming In 24 Hours, Greg Perry, Sams, 2001.

How Computer Programming Works, Dan Appleman, Apress, 2002.

Code: The Hidden Language Of Computer Hardware And Software, Charles Petzold, Microsoft Press, 2000.

Learning Computer Programming

Learning Computer Programming (with CD ROM), Mary Farrell, Charles River Media, 2002.

Computer Science Lab Home Learning Program (a step-by-step course for $19), www.computersciencelab.com.

e-Learning For Dummies (affordable on-line learning from the same people who brought you the books), www.dummies.com.

Microsoft Express (free hobbyist software), http://lab.msdn.microsoft.com/express/

Learning Programming Languages

Dr. Dobb's Journal (one of the oldest and most trusted sources of information on computer programming), www.ddj.com.

Programming In C (excellent free on-line tutorials to get you started), www.cprogramming.com.

Programming Tutorials (more free on-line tutorials on just about any computer programming language you can think of), www.programmingtutorials.com.

Association of C and C++ Users (even more free tutorials that feature excellent learning strategy sessions), www.accu.org/resources/public/terse/learn.htm.

Some Free Programming Software From Microsoft

Armed with this background, you can begin to learn to program in one or more languages. But just how do you do this without spending hours in a classroom or significant amounts of money on editor, compiler, or debugging software?

Well, fortunately, Mr. Bill Gates wants you to become a computer programmer and he wants to help you in some rather interesting ways. First off, Gates will provide you with working software that will allow you to program in popular languages such as Visual Basic, C++, and JAVA, and he'll provide you with on-line training for those languages as well. Best of all, it's all free!

Now, there are a couple of catches to this wonderful offer. First is that all the

software and documents are "betas" and, unfortunately, that means that you may run into bugs or incomplete features. The other catch is that these are not the "standard" editions that are used by professional programs, but "express" versions that are designed specifically for beginner or hobbyist programmers. However, this line of programming tools is precisely what you, as a beginner programmer, need because they will allow you to create usable software without having to work in a complicated environment.

Over the next several columns I'll focus on how to set up one of Mr. Gate's "Express" software language programming tools so you can do it yourself. Then I'll show you how to use that programming tool to add customized features to Flex Radio System's PowerSDR software in order to control the SDR-1000R.

Again, if you start with an overview of the basics and work your way through the primary concepts needed to understand how computer software programming is undertaken, the actual act of programming is not all that complicated.

So begin your "career" as a computer programmer by checking out one or more of the books listed in the sidebar. Remember, many of these books are available through your public library or can be purchased used in stores or though on-line booksellers such as Amazon.com.

Again, as I have stated in the past, do yourself a favor and keep a log of the work efforts you have made while learning computer programming. There's a lot of detail to be found in software programming, so when you learn something, write it down, otherwise you will lose it and have to re-discover it (if you are lucky). Make it a habit now and you will thank yourself in the years to come.

Next Month

Next month we'll see what's involved in downloading and installing one of the "Express" software development tools. We'll also take a look at the structure of the source code files that Flex Radio System provides for the PowerSDR software. Yes, it is a little complicated at first, but once I've shown you the layout and structure of the files and how they are viewed and modified in the editor program, you will begin to see the bigger picture of computer programming.

You can e-mail me with any questions you may have. Use my e-mail address at carm_popcomm@hotmail.com. As mentioned before, I cannot answer general questions on computers, but will be more than happy to help you with any issues raised in the columns.

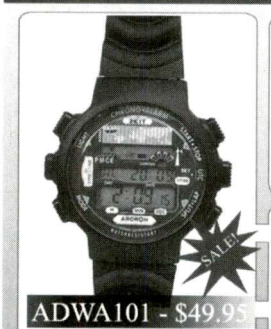
During the writing of this column Hurricanes Katrina and Rita had caused horrific devastation to New Orleans and many other communities along the Gulf Coast, and Hurricane Wilma blasted Florida. While these areas may be in the process of rebuilding as you are reading this, please do not forget that many people still need a great deal of help.

I would suggest you send donations to the American Red Cross (www.redcross.org/donate/donate.html). However, there are also many other good (and ethical) organizations that you can contribute to, so please use them if you wish.

Despite the many terrible things that have been happening at home, do remember our troops overseas and give them your support. As mentioned previously, the "Any Service Person" mail program no longer exists for security reasons. Please refer to the U.S. Department of Defense's official webpage, "Defend America." They have a specific section found at www.defendamerica.mil/support_troops.html that has an amazingly wide range of practical and useful ways that you can directly help.

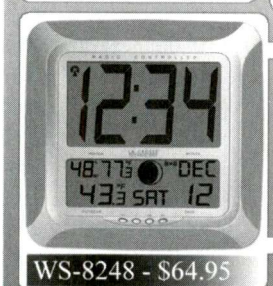
If you are fortunate to have a home, a job, and your loved ones around you in these times when so many don't, please remember to give thanks for your personal blessings and remember to pass that blessing on to others through sharing. See you again next month! ■

ATOMIC TIME
1010 Jorie Blvd. #332
Oak Brook, Il 60523
1-800-985-8463
www.atomictime.com



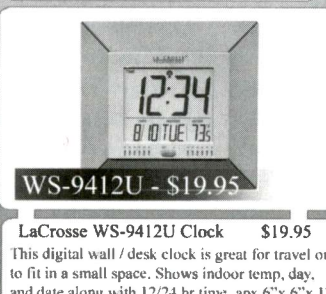
14" LaCrosse Black Wall
WT-3143A $26.95
This wall clock is great for an office, school, or home. It has a professional look, along with professional reliability. Features easy time zone buttons, just set the zone and go! Runs on 1 AA battery and has a safe plastic lens.

Digital Chronograph Watch
ADWA101 $49.95
Our feature packed Chrono-Alarm watch is now available for under $50! It has date and time alarms, stopwatch backlight, UTC time, and much more!



LaCrosse Digital Alarm
WS-8248U-A $64.95
This deluxe wall/desk clock features 4" tall easy to read digits. It also shows temperature, humidity, moon phase, month, day, and date. Also included is a remote thermometer for reading the outside temperature on the main unit. approx. 12" x 12" x 1.5"

1-800-985-8463
www.atomictime.com
Quantity discounts available!



LaCrosse WS-9412U Clock $19.95
This digital wall / desk clock is great for travel or to fit in a small space. Shows indoor temp, day, and date along with 12/24 hr time. apx 6"x 6"x 1"

Tell time by the U.S. Atomic Clock -The official U.S. time that governs ship movements, radio stations, space flights, and warplanes. With small radio receivers hidden inside our timepieces, they automatically synchronize to the U.S. Atomic Clock (which measures each second of time as 9,192,631,770 vibrations of a cesium 133 atom in a vacuum) and give time which is accurate to approx. 1 second every million years. Our timepieces even account automatically for daylight saving time, leap years, and leap seconds. $7.95 Shipping & Handling via UPS. (Rush available at additional cost) Call M-F 9-5 CST for our free catalog.

by Joe Cooper, carm_popcomm@hotmail.com

Programming Software Overview—Understanding Which Software Package Is Best For You

As I have pointed out in the columns that have been published over the past year, the "leading edge" of modern radio design has shifted from the use of "real" electronic parts to "virtual" components. These virtual components are created when specially designed software programs are run in a personal computer and, rather than tuning a signal as conventional radios do, they "process" radio signals as digital information.

Due to this significant change in radio technology many hobbyists who have enjoyed building radios in the past are now faced with a dilemma: how do we continue to enjoy "hands on" construction projects when radio circuits are now computer software algorithms? The answer is simple: if that's how radios are now being built, learn how to do that too using the new tools we have available.

Frankly the radio monitoring hobby has faced several major shifts in technology in the past and has managed to survive each one successfully. One example I'm old enough to remember was the big switch that took place between the use of vacuum tubes to transistors, and then to integrated circuits.

Even today there is still a strong contingent of people who proudly claim, "real radios glow in the dark," who take great pride in maintaining their vintage equipment in top form. Likewise, you'll still find a significant group of people who will only use vacuum tube based amplifiers in their high-end audio equipment.

What's really going to bring about the biggest changes in radio monitoring will be the new modes of digital radio transmission that will be taking over the airwaves soon, such as DAB (Digital Audio Broadcasting), DRM (Digital Radio Mondiale), HDC (Hi-Definition Compression) and more. AM and FM radio, which have been with us since the 1920s and 40s, will eventually be retired from use in commercial broadcasting, rendering all of those wonderful analog radios we now own obsolete.

So truly it's best to be ready for what's in store.

So in addition to learning how to use your computer hardware and operating system very well, the best way to be prepared for this new software-based technology is to learn how to write computer software programs.

As I've point out many times, it's actually not that difficult to learn how to be a software programmer, as long as you take your time, work out a goal and keep track of what you've learned.

Again, in last month's column I pointed out that it is not the programming languages that make computer programs work it's the sequence of concepts and the logic built into that software by the programmer that makes them truly functional.

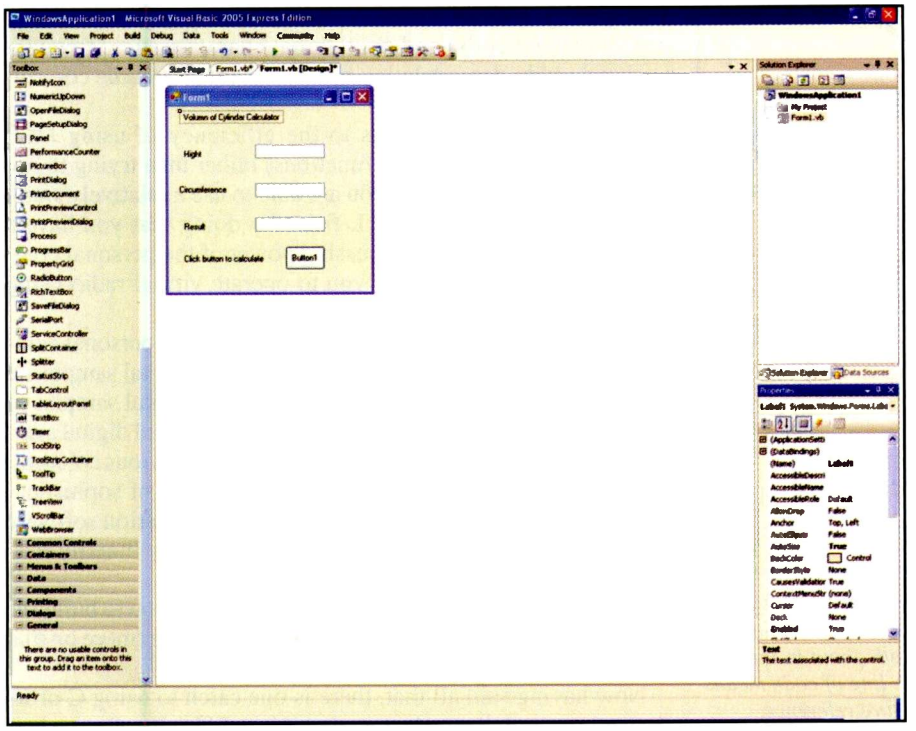
So what I'm going to be introducing you to in this month's column is the way in which you can create your new "workshop" for building Software-Defined Radios (not to mention other software projects that can be of direct benefit to the person who wants to perform computer-assisted radio monitoring). More importantly, what I'm also going to show you is how to set up this "workshop" for free, and that includes the training in how to use the software tools you choose to be part of your new toolkit. So let's get started.

Choosing The Proper "Workshop"

One of the biggest obstacles that most people face when sitting down to the task of learning how to become a software programmer is choosing which language to learn first. Frankly there are a large number to choose from, ranging from the ones used in commercial software programming (FORTRAN, COBOL, C+, BASIC, Visual BASIC, JAVA, to name a few) to ones used in more specialized situations (Euphoria, Smalltalk, Ruby and Python, for example).

Again, what is extremely important to understand is that the primary task that the computer programmer performs is translating human concepts into a form that a computer can "use." You must remember that ultimately the only program that a computer can "understand" is the binary code of 0 and 1 that make up "Machine Language." This set of instructions tells your computer's CPU (Central Processing Unit) what to do. In order for your computer to work, literally millions of 0's and 1's stream through the CPU every second, providing instructions for processing digital information.

Given such a vast number of 0s and 1s involved it's impossible for a human being to be able to write computer programs in Machine Code. The closest that we can get to



Welcome to your new workshop! This is where 21st Century radio hobbyists will build their software-defined radios and other computer-assisted radio monitoring devices. Pictured here is Microsoft Visual Basic 2005 Express edition, showing the cylinder volume calculator I talked about in this month's column.

writing directly in Machine Code is programming in Assembly Language, which uses grouped alphabet characters, called mnemonics, to represent binary code instructions.

Now when all is said and done, what you accomplish by writing a program in Assembly Language is a relatively simple program that takes a numerical value out of your storage memory (RAM), places it into the CPU for processing and then places the results in another memory location (again in RAM).

The results of using this type of programming is an extremely fast CPU processing of a large amount of information, which is important if you are doing a large number of mathematical calculations (say for when predicting weather for all of the continental United States for a week).

However, to be able to create such a program you need to have an extremely complex understanding of the computer hardware you are using and the information you are going to have it process. That task is going to involve a tremendous amount of time and energy, so such work is reserved for critical projects (like weather prediction, nuclear energy and physics experiments, to name a few applications).

So the vast majority of computer software programming used, "high-level" programming languages (such as FORTRAN, COBOL, C+, BASIC, Visual BASIC, JAVA, named before) due to the simple fact that they are relatively easier to work with than either Machine or Assembly languages.

More importantly, rather than simply performing "brute force" processing tasks that are performed using Assembly Languages, such as mathematical calculations, these high-level languages allow for more subtle concepts to be programmed, such as modeling a radio's electronic components in virtual form.

So when you choose a computer software language to perform that task of converting a radio into mathematical formulas, what you must first do is ask which programming language will best convey those modeling concepts in software form. Making a choice as to what language is appropriate for a particular programming situation is more of an art than science, and frankly for the types of situations that you will be encountering, only a handful of choices really need to be considered.

Likewise if you are interested in creating software for radio monitoring applications based upon the Microsoft Windows operating system, then you are really looking at two programming working environments—various versions of C (or more currently C, C+ and Microsoft Visual C++) and Microsoft Visual Basic.

Each of these software programming languages presents its own particular strengths and challenges, so let me introduce you to their essential characteristics and explain how each can be used to help you create useful programs for computer-assisted radio monitoring.

Intro To C, C+ And MS Visual C++

The C programming language was originally developed in 1970 by Ken Thompson and Denis Ritchie at AT&T Bell Labs for use on the UNIX operating system. It was originally used to write system software, which is one of the major components of an operating system that is used for running computer hardware.

Due to the fact that C was originally designed to work closely with computer hardware it was actually designed to be closer to assembly language than to a high-level language like BASIC and was once referred to as a medium-level language.

However, what made C different from assembly language was that rather than creating one executable file for an entire application (which those who remember early Windows or DOS based programs will be familiar with), C programs used a core program that accessed multiple libraries of functions stored in individual files.

You can see that structure if you use your Microsoft Explorer program to look at the content of a directory for most of today's software programs. If you do that you will see that there is one file called an "application" with one or more files called "application extensions" residing in that directory.

The application is written using what is called a "core language," which provides the main functions of the software program, and more importantly, makes use of those previously-mentioned extra files.

Those extra files are called libraries and they contain sub-programs that perform various functions that are useful for the operation of the main software program. Technically they are called "Dynamically Linked Libraries" and you will note that their file extension is "DLL" (with the result being they are often called "DLL Files").

These DLL files become useful due to their re-usability in more than one software program. So rather than having to program a lot of small details over and over again, a programmer can use a previously written DLL in order to save time.

How to say "Hello, world!" in a computer language.

Assembly Language

```
MODEL   SMALL
IDEAL
STACK   100H

DATASEG
        MSG DB 'Hello, world!', 13, '$'

CODESEG
Start:
        MOV AX, @data
        MOV DS, AX
        MOV DX, OFFSET MSG
        MOV AH, 09H       ; DOS: output ASCII$ string
        INT 21H
        MOV AX, 4C00H
        INT 21H
        END Start
```

C++

```
#include <iostream.h>

main()
{
    cout << "Hello World!";
    return 0;
}
```

Visual Basic

```
' A "Hello World!" program in Visual Basic.
Module Hello
   Sub Main()
      MsgBox("Hello World!")   ' Display message on computer screen.
   End Sub
End Module
```

One of the first computer programs that anyone learns is one that prints out "Hello, world!" on a video screen. Here is an example using the three computer programming languages that I talk about in the column. Don't worry about the details of what each line of code means right now. Just notice the differences for future reference.

In fact when you upgrade or add new features to a software program what you are most often doing is simply adding new DLL files to the library of an existing application.

Very often one software company may buy custom made DLL files from another software company that specializes in writing them in order to make their own software perform certain tasks, thus saving valuable programming time.

Likewise, many hobbyist software programmers share DLL files with one another for the same reason; so that people can add expanded functionality to their own software programs without having to spend time programming small details.

The power and flexibility of this type of programming was further enhanced during the 1980s as C became C+ (pronounced C Plus) and then C++ as new features came to be incorporated. These new features provided a very sophisticated programming method called "Object Oriented" (or OO), which models the characteristics of abstract or real things using objects.

Now to put this into terms that are understandable, an object is seen as having two basic characteristics: a state (of being) and (personal) behavior. For example, all pet dogs have a state (of being), that is they have a name, a color, a breed and they are hungry on a regular basis. Likewise the majority of pet dogs exhibit the same (personal) behavior, barking, fetching, and tail waging.

Even inanimate objects have a state (of being), such as the bicycle, which has gears, pedals, two wheels and motion. It even has (personal) behavior, such as braking, accelerating, slowing down and changing gears.

As you can see, once you are able to model something (e.g. small dogs + bark a lot = Jack Russell Terriers, or 10 speed + fast acceleration = racing bicycle) you are able to treat the results as having a "virtual" existence within your software program.

The key point here is that radios as objects have states and behavior too. (Have you experienced the big "ah-ha!" yet?)

So due to the fact that C+ is very close to assembly language it operates very fast with a personal computer, and because of that speed you can model the electronic components (resistors, capacitors and coils) and operational characteristics (frequency, bandwidth, demodulation mode and audio volume control) of a radio in virtual form.

More importantly (thanks to the efficiency of using DLL libraries to perform various functions) rather than trying to run one big software program, you are able to use a relatively small one thanks to the use of DLL files. By doing that you have a great deal of "left over" processing power of the personal computer's CPU that will allow you to operate virtual radio using SD radio software.

Again, you can operate a virtual radio within a personal computer by supplying the software program with digital samples of radio signals. This is accomplished by taking digital samples of radio signals off of an antenna by using a wide band digital sampling device, such as the ones I've described in previous columns.

Now if you want to take C+ to the next level of sophistication the current standard for programming application software for personal computers is C++, which is used by literally millions of computer programmers worldwide.

There is Microsoft Visual C++, which allows you to build the graphic user interface (or GUI) using simple tools (more on that in the next section on Visual Basic).

Now having said all that, there is one catch to using C or its derivatives, and that is the fact that it is a difficult software language for the beginner to learn due to its mid-range language characteristics.

A better choice for those who wish to become good amateur software programmers, and still have the potential for working with Object Oriented Programming (OOP), would be Microsoft's Visual Basic.

As the name implies, its roots lay in the popular (and easier to understand) BASIC programming language, which has a proven track record of use and acceptance by those who want an easier entrance into software development. (And if you are curious, yes there was an "A" and "B" programming language before C, but that's a story for another column.)

Intro To MS Visual Basic

The Beginners' All-purpose Symbolic Instruction Code (BASIC) is a computer language that has been around for 40 years now, having first been developed by mathematicians John Kemeny and Tom Kurtzasat at Dartmouth College in 1964.

BASIC is a high-level language that uses an English syntax like LET, PRINT, IF and GOTO as commands, and in earlier forms provided line numbers that allowed a programmer to keep track of a task they were working on.

While originally designed as a training tool, BASIC has the capability of being expanded for use in more sophisticated tasks. This capability was used by Bill Gates and his business partner Paul Allen to create Altair BASIC in 1975, which was Microsoft's first commercial product. (It was also Gates' BASICA that brought Microsoft into a partnership with IBM due to the fact that company's new IBM-PC needed a programming language. That opened the door for Gates to develop PC-DOS, and later MS-DOS and Windows, thus gaining control of the personal computer world for the next decade).

So again, due to its origins as a training tool, many people have come to feel that BASIC is not a "real" programming language like C++, despite the fact that it is used today in powerful products like Oracle Power Objects and Microsoft Visual Basic.

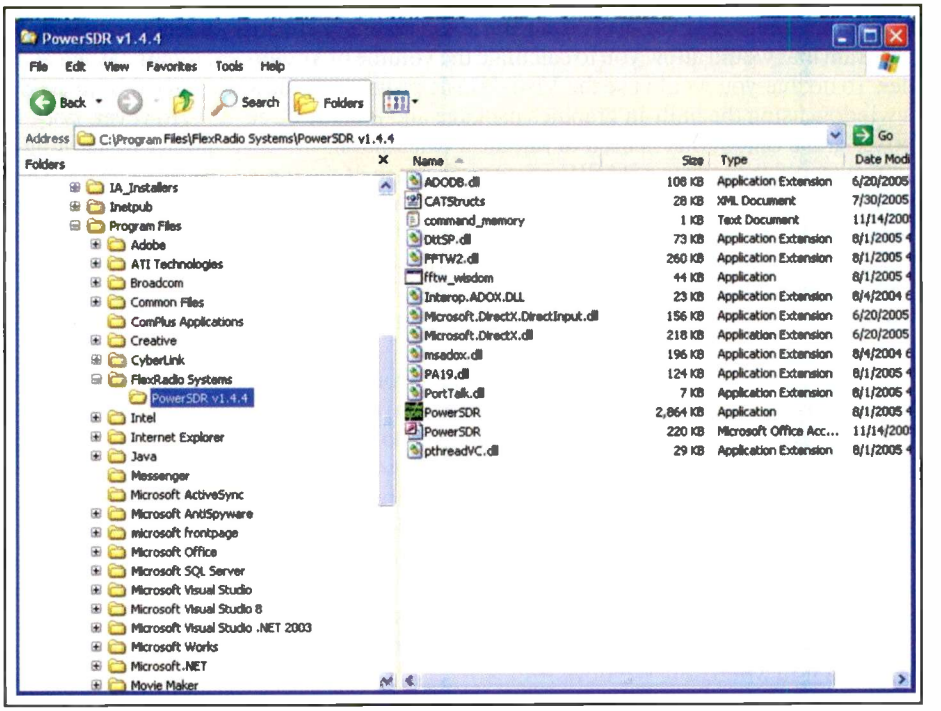
Visual Basic (or VB) was developed by Microsoft in 1987 as a direct competitor to C and C++, and while it was not initially accepted by the programming community, today it is the fastest growing programming language on the market.

The primary strength found in Visual Basic is that you can create software code in minutes that would normally take you days in C+. Visual BASIC does this by allowing programmers to attach software code written in a sophisticated version of the BASIC programming language directly to the graphic elements used in the Windows operating system, such as buttons, pull down menus and text boxes.

So in order to create a software program using Visual Basic, what you essentially do is perform three steps.

First you use a graphic designer in the VB programming software to draw a user's interface with all of the buttons and text boxes you require. Once that is done you then write the software code used to run the program and then "attach" that code to each of the buttons or boxes.

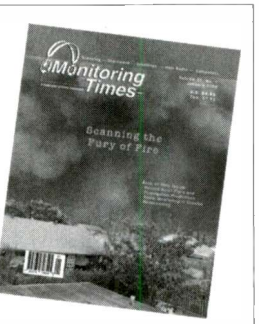
With all of the code attached, you then link the code together so that it performs the task that the software was designed to do. This can be illustrated by very simple example.



This is what happens after you write your code and create a finished software program. You end up with a mix of different types of files. As I outlined in the column, you have two primary types of files in a software application: one application file and many DLL files (plus a mix of additional files that provides documentation for configuring the application). Remember that the DLL files contain sub-programs used to reduce the overall size of the application by splitting it into smaller files. When you update a software application, these are the files that are replaced with new ones.

Monitoring Times

ORDER TODAY!



Do you own a radio, a shortwave receiver, a scanning receiver, or a ham radio? Then Monitoring Times® is your magazine! Open a copy of MT, and you will find news, information, and tips on getting more out of your radio listening. In fact, it's the most comprehensive radio hobby magazine in the U.S.

MT Express - Not only do we offer a premium monthly printed magazine, but we also offer the popular MT Express digital version of Monitoring Times. This is the best way to get your magazine as soon as it's released, right from the internet! All you have to have is an internet connection with an e-mail address and you're ready for all the excitement of a truly interactive publication. For less than the cost of a printed subscription, you get the full magazine in pdf format with **color** photos throughout and active links to all web sites mentioned in the text.

Packed with up-to-date information concisely written by the top writers in the field, Monitoring Times® is your foremost guide to profiles of broadcasting and communications installations; home projects; and tips on monitoring everything from air, sea, and space to distant ports of call.

Order TODAY and use code MTSPECIAL to get all the great offers!

7540 Hwy. 64 W.; Brasstown, NC 28902
1-800-438-8155 US and Can.; 828-837-9200; Fax 828- 837-2216
e-mail order@grove-ent.com

	6 months	One Year	Two Years	Three Years
US Rates	❒ $15.50	❒ $28.95	❒ $51.95	❒ $76.95
US 1st Class	❒ $30.00	❒ $57.95	❒ $112.00	❒ $168.00
Canada Surface*	❒ $20.50*	❒ $39.50*	❒ $75.95*	❒ $112.95*
Foreign International*	❒ $30.75*	❒ $59.50*	❒ $115.95*	❒ $175.95*
Electronic Subscription		❒ $19.95	❒ $38.90	❒ $57.85

*All payments must be in U.S. Funds drawn on a U.S. Bank!

Name _____ Address _____
City _____ State _____ Zip _____ - _____ Country _____
CC# _____ Exp. Date _____ CVV2 Code _____
Signature _____
Email address _____
MasterCard, Visa, and Discover Card accepted!

Or just go to www.grove-ent.com to order NOW!

GROVE
800-438-8155
828-837-9200 fax: 828-837-2216
WWW.GROVE-ENT.COM
7540 Highway 64 West
Brasstown, NC 28902

Say that you wanted to use Visual Basic to create a software program that would allow you to calculate the volume of a cylinder. To do that you would use the Visual Basic program to draw a window using the built-in graphics package and include three boxes and a button that is used to initiate a calculation.

You would then write a BASIC program for each of the three boxe—one to accept the height of the cylinder, another for the diameter and a third for displaying processed information.

Finally you would write a BASIC program for the button that would initiate the actual calculation when it was clicked (assuming that you had put numerical values into the first and second boxes).

That program would use the numerical data from the first to boxes to the formula for cylindrical volume ($\pi r2h$) and then display the product of that calculation in the third box.

Once you finished doing that you would write another BASIC program that would link together the three boxes and to that button, so that when you clicked on the button with your Window's mouse the calculation would be performed and the product displayed.

When you were finished with all of those tasks you would then turn what you had written into an executable program and then run it in Microsoft Windows.

The truth is that most advanced programming techniques comes from learning how to build such simple software programs and then learn to attach more and more features as you progress in your knowledge.

So for many people who want to become hobbyist computer programmers, particularly for computer-assisted radio monitoring projects, Microsoft Visual Basic is an ideal beginning point.

So, Which One Do You Use?

Now the question you might have is, "well you told me earlier that you need OOP software like C+ and C++ in order to be able to model a virtual radio. Can you do the same thing with Visual Basic?"

Fortunately the answer to that is yes.

With the introduction of Microsoft Visual Basic 6.0 in 1998, all of the key components required for OOP (abstraction, encapsulation, polymorphism, and inheritance, which I'll explain in detail in a future column) were included.

Since then each new version of MS Visual Basic (such as the 2005 version) have had further enhancements of OOP included, so as a result you can develop sophisticated software for Software-Defined Radios, or other computer-assisted radio monitoring applications, using this software development tool.

However, as previously mentioned, Microsoft also makes Visual C++, which allows you to create Windows software programs in the same way as MS Visual Basic, but using C++ code instead.

So given the choice between the two programming languages, which one is "the better choice"?

The truth is that today it's becoming "six of one, half dozen of the other" when it comes to comparing. Anything written using C++ will save programming time because you use libraries of pre-written code and you also save time with Visual Basic because you build a significant part of the program using graphic tools, rather than code.

However, the bottom line for most software programmers is the fact that C++ is closer to machine language in form than BASIC, so programs written in C++ will be faster. In that regard it's not surprising that for the best performance of virtual radio circuitry (such as is used in Software-Defined Radios) the language of choice is C++.

However, does that mean that you have to give up the simplicity of learning BASIC if you are new to software programming? The answer this time is no, because the refinements that have been made in the current versions of MS Visual Basic are making performance less and less an issue.

So the answer seems to be to try programming in both environments and see what you think of them. I'm sure your first reaction is "well, where am I going to find the money to spend on two pieces of software, one of which I may not like?"

Well thanks to the fact that Bill Gates wants you to become a software programmer you can download hobbyist versions of Microsoft Visual Basic and Visual C++ (along with several other programming languages) and use them for free!

Not only that, you can also get various types of training programs (ebooks, tutorials, on-line videos) for each of those programming languages directly from Microsoft for the same cost —free! This is an incredibly generous offer that you really should not pass up if you want to begin learning how to program in the Windows environment.

More importantly, there is a time limit on this free offer which will end on November 7, 2006. The thing to keep in mind is that while these are hobbyist versions of the programming languages, they are not toys. Not only can you use them to learn to write useful computer software by using them, you are also allowed under the license agreement that comes with them to create commercial (e.g. you can sell them for money) software packages without having to have a license from Microsoft.

So go to the Microsoft Express Editions at: http://msdn.microsoft.com/vstudio/express/default.aspx.

Take your time and read over the background information on the software that is offered there, which includes Visual BASIC and C++, along with several other important programming languages. Also make certain that your computer has enough hard drive space and operating power to be able to use the programming software properly. For best performance you should have the following computer configuration (or better):

- 1-GHz Pentium Processor
- Windows 2000 with service pack 4 or Windows XP with service pack 2
- 256 Meg of RAM (or better)
- 1.3 GB of available hard drive space

As with anything to do with personal computing, the more RAM, processing power and hard drive space you have the better your computer software will perform, and this particularly applies to using programming software.

What you will discover is that the processing required to actually create a software program (compiling) takes a lot of computing power and will tax whatever system you have to its limits if you don't have power to spare.

Likewise, given the size of the software programs that you are going to be downloading you will find that it will take a very long time if you are using a dial-up modem, so be forewarned.

The first thing you should do is to read the instructional material provided on the Microsoft Express website, particularly the Frequently Asked Questions (FAQ), overview information and

feature tour. That information will allow you to make the right decisions about downloading and installing the software. More importantly, it will also point you in the right direction for learning how to program and connect to on-line communities of people, called forums, who are sharing information on how to use programming software.

The key point is that the quicker you begin to use programming software, particularly if you are doing so while following some type of structured training instruction like that provided by Microsoft, the quicker you will be able to master the skills involved.

As I have always suggested in the past, set reasonable goals for yourself, work at a short but consistent pace (such as an hour a day maximum) and keep a log of what you do with your computer, particularly to keep track of new programming techniques you have just learned.

If you do that consistently then in a surprisingly short period of time you'll find that you are making fewer mistakes and the code that you write will actually produce usable computer software.

Remember as well to enjoy the journey, be proud of your accomplishments as this is a challenging task, and remember that there are real rewards waiting for those who stay with that task to the end.

Next Month

I will be continuing this examination of computer programming, looking at what you get once you download and install one of Mr. Gates' "Express" software development tools.

Yes, it is a little complicated at first, but once I have shown you the layout and structure of the files and how they are viewed and modified in the editor program, you will begin to see the bigger picture of computer programming very quickly.

Do not forget that if you wish to e-mail me with any questions use my e-mail address: carm_popcomm@hotmail.com. As mentioned before, I cannot answer general questions on computers, but will be more than happy to help you with any issues raised in the columns.

Unfortunately the unbelievably terrible storms and hurricanes have continued to hit parts of the United States, including Florida and Indiana. What a year it has been with so many people having to face so many personal hardships. In saying that I would like to suggest that you send donations to the American Red Cross—http://www.redcross.org/donate/donate.html in order to help your fellow American citizens in this time of trouble. There are many other good (and ethical) organizations that you can contribute to, so please use them if you wish.

Once again, do remember our troops overseas and give them your support. As mentioned previously, the "Any Service Person" mail program no longer exists for security reasons. Please refer to the U.S. Department of Defence's official webpage, "Defend America." They have a specific section found at: to www.defendamerica.mil/support_troops.html and it has an amazingly wide range of practical and useful ways that you can directly help.

If you are fortunate to have a home, a job and your loved ones around you in these times when so many don't, please remember to give thanks for your personal blessings by remembering to pass on that blessing to others through regular acts of selfless sharing. ■

INFOCENTRAL *(from page 5)*

the Thai service, the BBC is also expected to end its central European language services such as Polish, Czech, Slovak, Romanian, Hungarian and Macedonian. *The Nation* also reports that the BBC also wanted to shelve the Tamil and Nepalese services, but that was put on hold as the political instability in Sri Lanka and Nepal remained on the international news agenda.

American Military Launch Radio Rasul In Afghanistan

Radio Rasul, an Arabic word meaning "message," is a mobile radio station operated by the 82nd's 1st Battalion, 325th Airborne Infantry Regiment and the 44th Signal Battalion that broadcasts music, news, public service announcements and other information provided by the Wardag provincial government to the local population. The radio station gives the government a way to promote itself and its programs to the local people, the majority of whom don't read and don't have access to televisions, said American Army Lt. Col. David Anders, Commander, 1st Battalion, 325th Airborne Infantry Regiment.

The radio station made its first broadcast September 16 and has been running from 8 a.m. to 9 p.m. every day since then. It broadcasts over FM and has a range of 40 to 60 km. The station is totally mobile and can be used anywhere as long as it has a power source. The programming is pre-recorded and is then beamed out over the airwaves in a continuous loop.

SLBC Staff Protest Against Chairman

Staff at the Sri Lanka Broadcasting Corporation (SLBC) has started protests against SLBC Chairman Hudson Samarasinghe. It's being alleged that Mr. Samarasinghe assaulted SLBC news director Raja Katugampola and threatened his life for broadcasting Premier Mahinda Rajapaksa's election campaign news. Mr. Samarasinghe was admitted to the hospital following the alleged assault. SLBC staff has accused Mr. Samarasinghe of creating "terror-work conditions" at the SLBC, and are urging President Kumaratunga to dismiss him immediately.

A trade union spokesperson said that the Chairman "tries to introduce dictatorship" in SLBC. The Sri Lanka Progressive Front trade union is angry with Mr. Samarasinghe for "crippling" the ruling party presidential candidate's election campaign.

Japanese Abduction Inquiry Group To Start Shortwave Broadcasts

Kazuhiro Araki, head of the Investigation Commission on Missing Japanese Probably Related to North Korea, has told reporters that the group plans to start shortwave broadcasts. The broadcasts, in Japanese, will be beamed to North Korea in the hope of reaching abducted Japanese citizens, who the Commission hopes will respond to the messages. No details were given of the times and frequencies, and whether these will be self-contained broadcasts or inserts into existing Japanese transmissions. (It seems unlikely that there will be any response, given that listening to shortwave broadcasts from overseas is forbidden in North Korea, and in any case making contact would be extremely difficult and dangerous). ■

by Joe Cooper, carm_popcomm@hotmail.com

Learning Microsoft Visual Basic—How To Build A CAT Program For Ten-Tec's RX-320D

As followers of this column now know, affordable high-speed PCs make it possible to take a radio signal and turn it into digital information. This digital information is then processed through a "virtual" radio created by computer software and indistinguishable in its performance from a "real radio." So rather than creating "real" radio circuits with wires, circuit boards, and traditional components, software programmers now create those electronic parts in "virtual" form by representing them as a mathematical formula.

As a result, the shop bench of the days of analog radio, with its soldering iron, tools, and test equipment, is being replaced by a PC loaded with programming software. Rather than bringing about the end of the radio monitoring hobby, the PC has actually created an exciting *new* opportunity for the radio-monitoring hobby to revitalize itself, particularly for those people who like to build their own equipment. The challenge is to learn how to write software programs that will create "virtual" radios in their computers.

Taking Up The Programming Challenge

So if you want to learn how to create a virtual radio through software programming, how do you get started. The last two columns have outlined how to begin to do just that, and last month I introduced you to two very important programming languages that can be used to create applications for running virtual radios, Microsoft Visual C++ and Visual Basic.

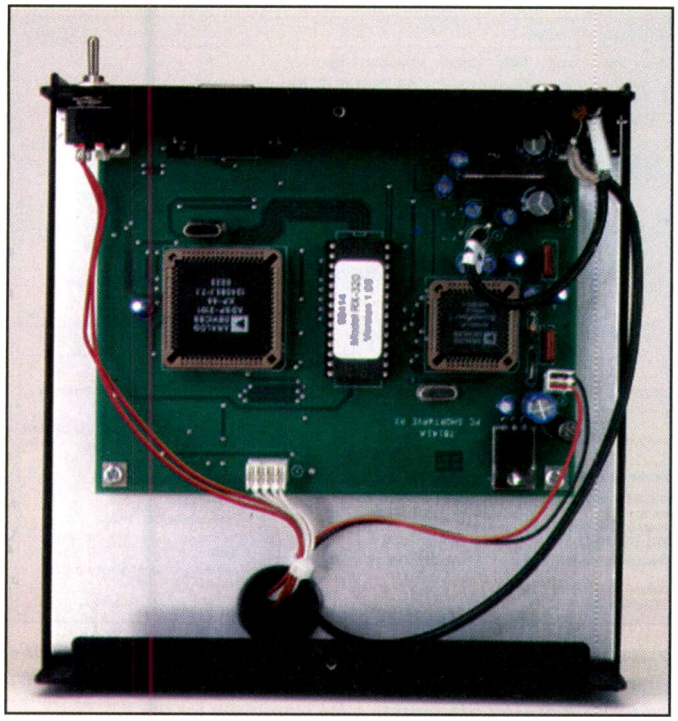
I focused on these products because Microsoft is offering them, along with online training, for free for a one-year period through the Microsoft Express program (http://msdn.microsoft.com/vstudio/express/default.aspx), designed to get more people involved in computer programming at a hobbyist level. While the programs are geared toward those who want to learn computer programming, the software itself is the "real thing," only without some of the bells and whistles found in the complete package.

So given the great learning experience Microsoft is offering at no cost—other than your own computer and time investment—this is a wonderful opportunity to master an increasingly valuable set of skills (not to mention start a new career if you are so inclined).

To help you along I am going to provide you with an introduction to the software programming environment of Microsoft's Visual Basic. There is a bit of a learning curve between a beginner's level of programming knowledge and that required to create a fully operational SD radio. This doesn't mean that the task is impossible or impractical, but it needs to be tackled step-by-step so each new skill you learn becomes the foundation for the next.

First you should begin with a simpler software development project that will provide you with proper foundation and serve as a bridge to more complex software programming. Interestingly, our first software design and development project will be aimed at one of the earliest, and most successful, Software-Defined Radios (SDRs), which is still offered to the general public: Ten-Tec's computer-controlled RX-320D

Ten-Tec's RX-320D has been around for a number of years now, but is still one of the best examples of good SDR technology on the market. Unlike more contemporary designs, all the processing takes place in the "black box," rather than in a host computer. All the functions (tuning, volume control, and so on) are done through an attached computer via the serial cable. (Photo courtesy Ten-Tec)



Inside the RX-320D are three chips; the primary one, the ADSP-2101 built by Analog Devices, is on the left. This is actually a "computer on a chip," and uses computer code stored on an EPROM (the middle chip). The chip on the right connects the main IC to the computer containing the control codes, which provides values for functions like volume control and tuning. (Photo courtesy Ten-Tec)

(http://radio.tentec.com/Amateur/Receivers/TT320). I'll focus on a project that will create a Computer-Aided Tuning (CAT) software program for the RX-320D using Visual Basic. While this project is not the same as creating true SDR software (which we will do in a future column), it's a very important beginning point for learning how to design and build your own application software using a programming language like Basic.

Intro To CAT Software

Back in the March and April 2004 issues of *Pop'Comm* I covered Ten-Tec's RX-320D "black box" HF receiver in full detail, where I first introduced SDR technology. This radio was one of the first SDRs on the market and its design is notable because all the DSP (DSP) takes place in the radio itself through a built-in computer on a chip, rather than having to be processed in an external computer. Because it has no external controls, other than an on/off power switch, you have to connect the radio to a PC via a serial cable to operate it using a CAT software program.

To design a computer software application you must first understand how the RX-320D operates, which is very different from most conventional radios you may be using. The first clue to that difference comes when you take off the cover and look inside. Rather than a conventional collection of circuit boards and multiple components, you see three integrated circuits (ICs). The circuit that does most of the work is actually a small computer, not unlike the PC you connect to the radio in order to control it. The chip is an ADSP-2101, manufactured by a semiconductor company called Analog Devices (www.analog.com), and it defines the virtual radio circuits used in DSP applications.

Table 1. Command Codes

Function	Command Code Variable
Mode	AM, USB, LSB, CW
Frequency	100 kHz to 30 MHz
BFO Offset Frequency	0–2000 Hz
Audio Filter	300 Cycles to 8000 Cycles
AGC Control	Slow/Medium/Fast
Line-in Level	0–63
Speaker Output Level	0–63

The software used to define those virtual circuits was written by the radio engineer who designed the radio itself. It's stored in a semi-permanent state, called Firmware, on a special memory chip, called an EPROM, which stands for Erasable Programmable Read Only Memory. While this is all very good for creating the "innards" of the radio, remember that there are no controls on this black box, so you'll need a second piece of software, known as either "control software" or "CAT" software, that you run on your PC.

CAT software sends (and sometimes receives) *command codes* that are used to change the setting of the virtual components in an SDR. So if you want to change the frequency, control the audio volume, or set the operating mode, you need to send a command code to the computer in the radio to operate these virtual controls. The command codes used in the operation of the RX-320D are a set of seven DSP functions that the ADSP-2101 chip is programmed to perform. See the accompanying **Table 1** for a list of these functions.

The command set also controls two requests for information that the ADSP–2101 can respond to with information that can be displayed on your computer screen. These are:

Request	Response
Signal Strength	0 –10,000
Firmware Version	VER XXX, where "X" equals a numeric value.

Command codes are actually very simple, being a combination of a letter, a number, and a carriage return. Each function has its own letter to represent it (e.g. the letter "M" for Mode) and a number to represent either a fixed function (M0 = AM mode and M1 = USB mode) or a range of numbers.

When the control software on your PC is operating and connected to the RX-320D via its serial cable, it's the command codes that are being passed to the radio. Likewise, when a request for the signal strength is made, the response comes back from the radio via the serial cable to the PC and is displayed there in the control software user interface (what you see on the computer screen when you run the program).

But how exactly does that command code get generated, then directed into serial cable, within the PC? That's where the CAT software comes in. It provides you with a simple way of generating that code and sending it off to the radio and keeps you informed about the general operating status of the radio at any given time.

Using Visual Basic, we'll examine the individual components of a software programming language. While it may sound intimidating, once you begin to see each line of code in context and within the bigger picture of the working software program as a whole, it's not as difficult to understand as it first appears. As

with any project, begin with a goal, a plan, and a design before you begin to write any code in any programming language.

Designing A CAT Software Program

To succeed in creating your own computer software, the first thing you need is a notebook for keeping a record of your project. Software programming requires that you document what you're planning and track the tasks you've performed throughout the project. Nobody has ever created a truly successful software program by simply sitting down in front of programming software and typing until it's done.

The beginning point of any software program construction project is a clear goal for the project and a mission statement. In our particular project the mission statement can be:

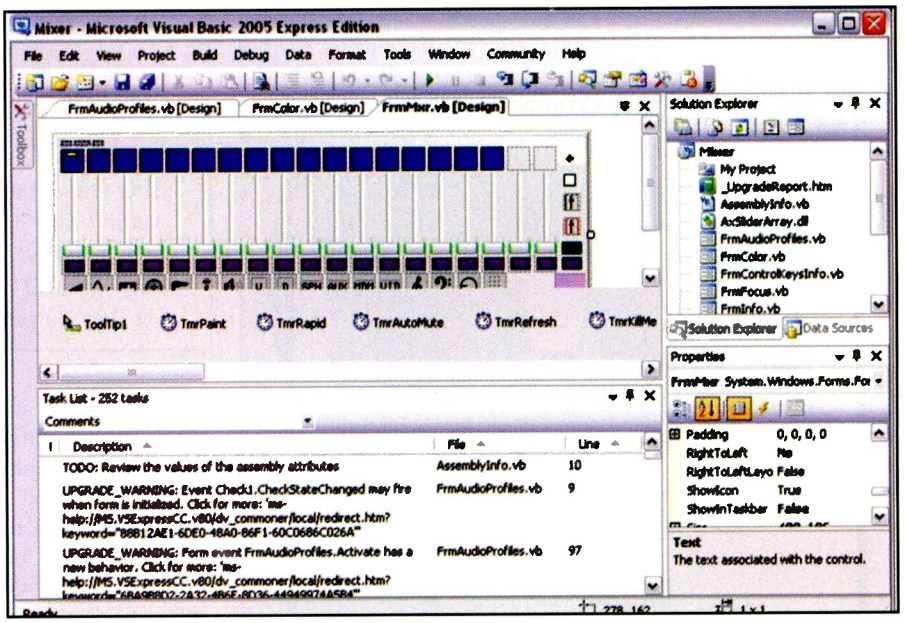
"To successfully control the seven main features provided for the operation of Ten-Tec's RX-320D and provide a simple and easy-to-understand user interface when doing so."

You can then go on to list what those seven features will be, as follows:

- Be able to tune AM, USB, LSB, CW signals
- Provide a frequency tuning range of 100 kHz to 30 MHz
- Provide a BFO offset frequency of between 0–2000 Hz
- Have a multiple step audio filter that ranges between 300 cycles to 8000 cycles
- Provide AGC Control in Slow/Medium/Fast settings
- Provide line-in level
- Provide speaker output level

You want to list everything because later on you'll need to create a checklist to ensure that everything you want to include in your software design is in place. This may seem easy in this project, but some software programs may include hundreds of features, with the development of each delegated to different software programmers. Even if it's only you working on this project, you can still lose track of where you are, so documentation is critical.

Next you need to work out the actual tasks your software will perform so you have a clear idea of the correct order for your software programming steps. You have a fair degree of latitude in exactly



Next month we'll begin to look at your new work bench: Microsoft Visual Basic Express. It's a little intimidating at first, but there are built-in training programs that come with this application to help you tackle a project like writing a CAT program for the RX-320D.

what takes place (after all, it is *your* software and *your* radio), but the key point is that you want things to take place in a fairly sensible order.

Imagine that you have a radio in front of you, and ask yourself what do you have to do to operate it and what do you want to happen as you do operate it. Obviously when you turn on a radio you don't want the audio to come blasting out at high volume. Likewise, you don't want it to start in CW mode when you prefer to listen to voice transmissions. Do you want the radio's frequencies to start at the top of the frequency range so you can tune down, or at the bottom so you can tune up? Sit down and work out the default settings you want the radio to start out in when you "turn it on" using your software.

After you have determined the features and default settings, your next task is to design the user interface; that is, what you'll actually see on the computer screen when you use the software. Very simply, how do you want the "control panel" for the radio to appear? Do you want it to look like a conventional radio, with knobs and dials and tuning bar like on a radio from the 1950s? Or do you want something totally experimental and different that it no longer looks like anything someone could call "a radio" if they looked at it?

Visual Basic provides many graphic tools so you can create many different types of buttons, click boxes, display areas and input fields. Although you can do some final "tweaking" on the screen as you create the program, make sure you work out what you want your control panel to look like on paper rather than on the computer screen. Remember, you should always have most of your work tasks written down ahead of time, and that includes your graphical designs. This way you'll have less of a tendency to get lost in the code or end up with so many bugs in that code that it becomes a frustrating process simply making it work.

Writing Computer Code

So let's say you've written a list of the main features and functions to be included in your computer program, and you've designed how the graphical control panel for the radio will look on the computer screen. Your next task, which is really the "heart and soul" of computer programming, is to document the key concepts to be incorporated into the final software application. Here you really need to sit down and work out a full description of what the software will do when it is actually run. It should read something like this:

"When I turn on the software, it will first send all of the default values needed to turn on the RX-320D, set it to a particular frequency and mode, and keep the audio volume from blasting out of the loud speaker. I will then be able to tune the radio to a particular frequency, and I will do that by..."

You cannot successfully write a software application program without having a roadmap to follow. It's impossible to sit down and begin to write software from scratch. And, frankly, anyone who tries this approach is simply "hacking code," a term used to describe the act of taking a previously written program (generally developed by someone with better programming skills) and modifying it for a new application.

Work out the key functions for which you will need to write computer code and you will simplify your code writing tasks considerably. Once you have a clear idea of what you want to do, it's relatively simple to translate your human concepts into machine concepts by translating them into a language computers can understand. Learning a computer language is not that much different from learning a human one because, in the end, computer languages are simply words and grammar used to describe and explain things taking place within a computer.

Key points to document are:

- What will the default settings be when you start your application software?
- What will happen when you click on a particular button shown on the control panel?
- What "events" do you want to take place when certain things happen, such as when a series of buttons is clicked?
- What do you want to take place within the radio when the software is used?
- How will the computer "talk" to the radio so information can be passed back and forth?

In the case of the RX-320D, I determined that there are nine things that the software needs to do in order to operate the radio properly. These are:

- Set the default values for operating the radio
- Open the serial cable connection between the computer and the radio
- Run the routine functions of the software so that all the radio functions work
- Tune the radio
- Present an overall design of the control panel as it is displayed on the computer screen
- Display proper information on the control panel (frequency, volume level, signal level, etc.)
- Process input values (frequency, volume level, mode, etc.)
- Send control codes to the radio
- Set default data values (such as values for the audio filter)

Rather than trying to write computer code as one big task, break down each of those tasks based on the list that you create, such as the one shown above. You'll find that when you break down these tasks into their individual parts they are actually very simple to program.

So, for example, using the original Basic programming for the RX-320D as an example, setting the default values would be done as follow:

```
AMMODE=ASC("0"):CWMODE=ASC("3"):USB-
MODE=ASC("1"):LSBMODE=ASC("2") ' modes defined
  DIM FILTERS(34)     ' array to hold filter list
  GOSUB 1360          ' preload filters array
  D$=CHR$(13)         ' carriage return defined
  SPKVOL=30           ' speaker volume
  MODE=AMMODE         ' detection mode
  RADIOFREQ=.93       ' tuned frequency
  FILTER=0            ' filter number from filter list
  CWBFO=0
```

All this really says is that when the radio is turned on it is set to AM mode, the audio filter is set to 10,000 kHz wide, the speaker volume is just below the halfway point and the frequency is set to 930 kHz. To place those values into the radio, you need to open up the serial port for communication between the computer and the radio. The command for this would be:

```
OPEN "com1:1200,n,8,1,rs,ds" AS #2
```

Most of these software commands are very common and easy to use, so finding the appropriate ones for your particular software application is not that difficult.

If you already own a computer-controllable radio other than the Ten-Tec RX-320D and you have access to the control codes (generally given in the user guide that came with the radio), why not sit down and begin to work out a design of your own? Even if you don't intend to learn how to become a software programmer, you'll certainly learn a lot more about your radio's operation than you knew before. You'll also begin to appreciate the engineering detail that went into your radio as you break down its functions in an organized way. Have fun trying out different designs for a user interface, and you may even come up with a unique layout that's never been tried before.

And who knows, even if you were a bit intimidated by the idea of software programming, you may find yourself gaining more confidence and motivation by doing these small steps. Maybe you'll end up actually trying some software programming by following the instructions in next month's column.

Next Month

For next month's column, make sure you've downloaded and installed the free copy of Microsoft Visual Basic available at http://msdn.microsoft.com/vstudio/express/vb/default.aspx. Take some time to go through the free training in computer programming that's also provided there so you'll be ready for the next step in this project. I'll show you how to take all the documentation and planning that was done in this month's column and turn it into a working software application program.

Don't forget that if you want to e-mail me with any questions use my e-mail address: carm_popcomm@hotmail.com. As mentioned before, I cannot answer general questions on computers, but will be more than happy to help you with any issues raised in the columns.

While the hurricane and tornado seasons are now over, it's clear that we will need to be prepared for next year's crop. In saying that I would like to suggest that you send donations to the American Red Cross (www.redcross.org/donate/donate. html) to help your fellow Americans in this time of trouble. However, there are many other good (and ethical) organizations you can contribute to, so please use them.

As always, do remember our troops overseas and continue to give them your support. Please refer to the U.S. Department of Defense's official webpage, "Defend America." It has a specific section found at www.defendamerica.mil/support_troops.html with an amazingly wide range of practical and useful ways you can directly help.

Again, if you are fortunate to have a safe and secure home, a paying job, and your loved ones around you in these times when so many don't, please remember to give thanks for your personal blessings by remembering to pass on that blessing to others through regular acts of selfless sharing. See you again next month! ■

by Joe Cooper, carm_popcomm@hotmail.com

Building A CAT Program For Ten-Tec's RX-320

Thanks to the availability of inexpensive high-speed personal computers, it's now possible to create virtual electronic components using computer software. This month, we'll continue to outline how you can build virtual radios using Microsoft's new free Visual BASIC and Ten-Tec's RX-320 as the foundation.

As we've been discovering, here's no longer any advantage in building a complete radio out of "real" components today since virtual electronic components will out-perform real components every time because they always deliver 100 percent of their rated value, no matter what conditions they operate within. Plus these digital components cost far less to "manufacture" in their virtual form than real components.

Consider, for instance, that the Ten-Tec RX-320D has 34 selectable IF bandwidth filters ranging from 300 Hz to 8,000 Hz. To build a mechanical device with capacitors, coils, and crystals to reproduce the same filtering capability would be next to impossible from an engineering standpoint, and prohibitively expensive even if you could actually built it.

These two characteristics of virtual electronic components (essentially perfect-state operation all the time plus significantly reduced cost) motivated radio electronics manufacturing companies to create the first generation of DSP radios during the 1980s. The trend really took off when the first "computers-on-a-chip" ICs became available in affordable quantities.

Photo A. The Harris RF-5000 series of DSP radios from the late 1980s represented the pioneering era of software-defined radio (SDR). This is the RF-5020, which as you can see was built tough and represented a real departure from earlier military design by incorporating LCD display and push button operation.

SDR Radio And The U.S. Military

A good example of the breakthrough in digital radio design was the Harris RF-5000 military radio (**Photo A**). Introduced in 1988, it offered a unique design that featured one of the first DSP IF and internal CPUs on a card for computer control of its functions. That generation of radios was based on crossover technology—a digital radio emulating analog functions—but it was essentially no different in its general operation than the solid-state radios of the 1970s. Engineers of that time, however, foresaw that these radios could be made fully digital, allowing the creation of the first true software-defined radio (SDR) technology, and more importantly, the networking of the radios together via built-in computer technology.

Development intensified when the U.S. military realized the high level of reliability and performance the DSP radios displayed in the field. The confidence that the field troops and commanders displayed in their digital signal processing (DSP)-based radios convinced the U.S. military command, along with most of the military leaders of the developed world, to begin converting their entire communication system over to SDR technology as soon as possible.

This changeover from individual DSP-based radio systems to true SDR technology occurred during the early 1990s when project SPEAKeasy (Phases I and II) was initiated by the U.S. military in order to create a standard for SDR technology that would last well into the 21st Century. The goal of SPEAKeasy Phase I was challenging: the ability to communicate over 10 different types of radios found within all branches of the American armed forces and to do so as a single communications system over a frequency range of 2 MHz to 2 GHz. That phase was fully tested by 1997 and, while many "bugs" existed in the system, the basic concepts of SDR were found to work.

Further refinement took place during Phase II, the goal of which was to design and build a true SDR that would act as the backbone for the entire system. The criteria was that the resulting radio had to be small and portable, use open architecture software (e.g., easily upgraded with software created by third party developers), have full cross-channel connectivity (e.g., multiple modes of communication), and operate between 4 to 400 MHz.

The Phase II project was so successful that the research was fully completed in only 15 months (three years had been allocated) and the resulting radio technology went directly into production. It is currently being distributed by the Joint Tactical Radio System (JTRS) program.

Hobbyist SDR By Ten-Tec

At the hobbyist level, we began to see the benefits of newly developed "black box" radios during the late 1990s as the first

Table 1.

Function	Command Code Variable
Mode	AM, USB, LSB, CW
Frequency	100 kHz to 30 MHz
BFO offset frequency	0–2000 Hz
Audio Filter	300 cycles–8000 cycles
AGC Control	Slow/Medium/Fast
Line-in Level	0–63
Speaker Output Level	0–63

civilian SDR radios began to hit the store shelves. Ten-Tec's first SDR black box, the RX-320, was first offered in 1998 and was recently updated to the RX-320D, which is capable of receiving the new shortwave digital mode, called DRM (Digital Radio Mondiale).

This radio was one of the first SDRs on the market and its design is notable because all digital signal processing takes place in the radio itself through a built-in computer on a chip, rather than having to be processed in an external computer. Furthermore, it has no external controls, other than an on/off power switch, and you have to connect the radio to a PC via a serial cable to operate it using a CAT (computer-assisted tuning) software program.

The RX-320D is an excellent SDR computer to begin with because of Ten-Tec's "open source" philosophy regarding sharing information about the command codes the CAT software needs in order to operate the radio. CAT software sends, and sometimes receives, unique command codes that are used to change the setting of the virtual components within the RX-320D. So if you want to change the frequency, control the audio volume, or set the operating mode, you need to send a command code to the computer in the radio in order to operate these virtual controls.

The command codes used in the operation of the RX320D are comprised of a set of seven DSP functions that the radio's CPU chip (the ADSP-2101 "computer on a chip" by Analog Devices) uses to perform various tasks (see **Table 1**). The command set also controls two requests for information that the ADSP–2101 can respond to with information that can be displayed on your computer screen. These are:

Request Response
Signal Strength 0 – 10,000
Firmware Version VER XXX, where "X" equals a numeric value.

With those command codes as a foundation, you need to design a software program that can convey those command codes into the radio's CPU via the serial cable. Ten-Tec has provided all the required key information on their website (http://radio.tentec.com/Amateur/Receivers/TT320/Downloads) and you can download the software programmer's reference manual, which contains all of the command codes and example programming code in BASIC for MS DOS. Our task is then to transfer that information into a new software program based upon Microsoft's new Visual BASIC Express program, available for free at http://msdn.microsoft.com/vstudio/express/vb/ default.aspx.

Right now let's take a look at getting started with Visual BASIC Express and how to do the necessary preliminary planning before you begin writing code for the RX-320D CAT program.

Getting Started With Visual Basic Express

I've focused on Microsoft Visual BASIC because it is being offered for free, along with an extensive on-line training program, for a one-year period. This offer was made to allow as many people as possible to become involved in computer programming at a hobbyist level. While other groups and organizations previously offered free training programs, and even programming software, most of those programs were in non-standard programming languages or were geared toward a higher level, resulting in the industry being dominated by the proverbial "computer geek."

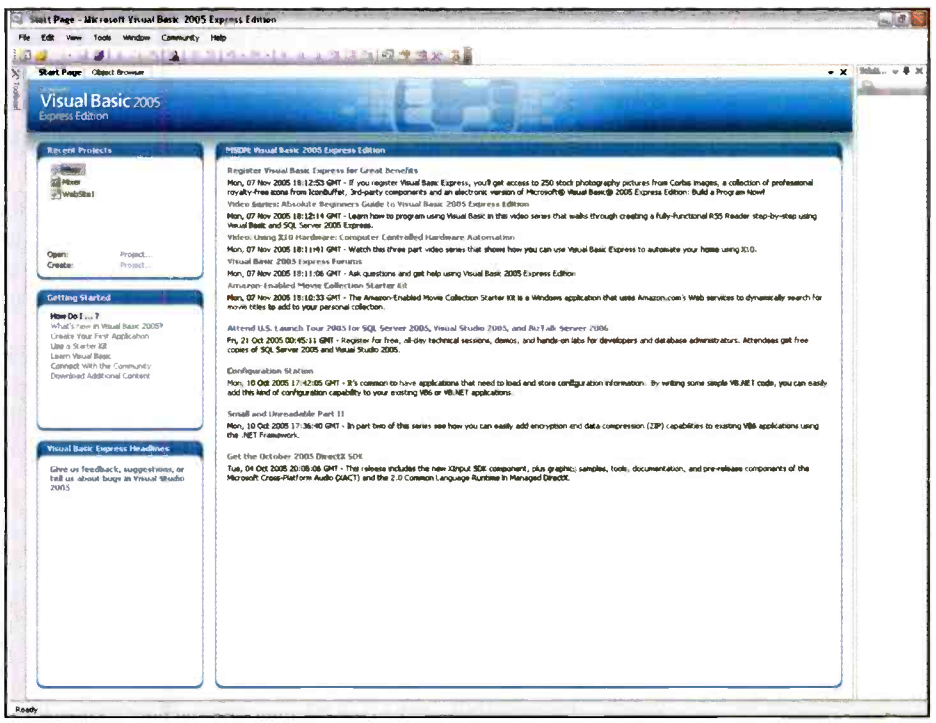


Photo B. When you begin using Microsoft's Visual Basic 2005 Express version, your starting point is a "Getting Started" tutorial. If you want to be successful in computer programming, just work your way through the exercises and soon you'll be "laying down code."

But now, thanks to the availability of cheap, extremely powerful PCs and the Internet, everyone who is "plugged in" has access to a huge library of information about computer programming. Today's computer industry is welcoming more people from more diverse backgrounds, and the biggest problem has become convincing the general public that computers are no longer overly complicated, particularly when it comes to computer programming. This has led to initiatives like Microsoft's Express program that can appeal to anyone who wants to become creative with his or her computer.

The Express software Microsoft is offering is not a toy, but the real thing,

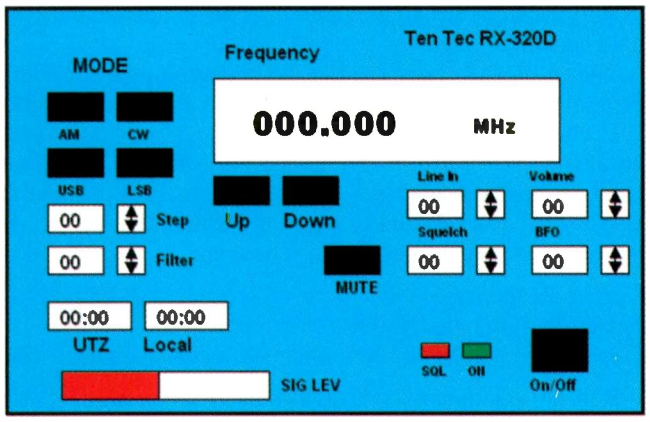


Photo C. This is the prototype computer user interface that I designed for the RX-320D software project for this column. It was created using Microsoft Word 2000, so you really don't need a fancy graphics program to follow this exercise.

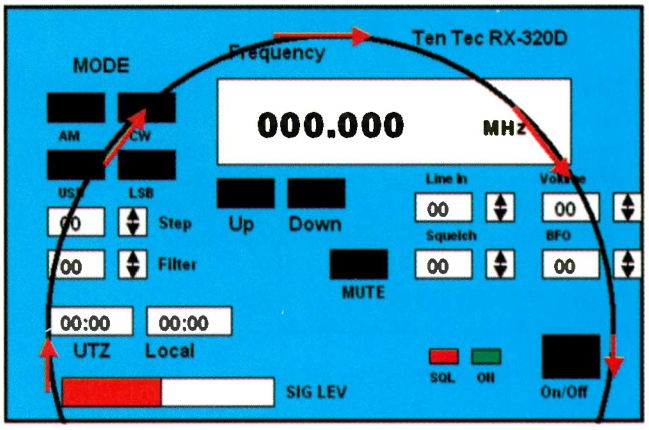


Photo D. This is the prototype design showing the viewing arch going from left to right. Note the way in which buttons and functions are grouped together to make access to them as easy as possible.

only with some non-essential bells and whistles found in the professional package missing. Given the great learning experience Microsoft is offering at no cost (other than your own computer and time invested in learning), this is a wonderful opportunity to master an increasingly valuable skill set that will directly benefit your radio monitoring hobby (not to mention maybe being the start of a new career if you're so inclined).

First Steps With Visual BASIC

In last month's column I suggested that you download and install Visual BASIC Express if you were sincerely interested in learning how to create a CAT program for the RX-320D. Hopefully, you've done that by now and have availed yourself of the free training provided. Frankly it's practically impossible not to be able to get started with this product given the way it's set up. When you open the program, the first thing displayed is a "Getting Started" screen that provides you with a step-by-step walk-through on using the program.

There are several tutorials (**Photo B**) and pre-built templates with which you can become familiar with the mechanics of using Visual BASIC, and you should create several working software applications using those tutorials before undertaking the project I'll be describing here. While I can't outline everything you should know to successfully program in Visual BASIC, I can point you toward the tools and give you a roadmap on how to increase your chances of success in this project. Remember, as in any human pursuit, there is no guarantee of success, and in the end your level of success is determined by your level of ability and willingness to apply yourself in achieving a goal. However, in the end, the only real failures in life take place when one does not even try to do something out of a fear of failure. So even if you don't achieve 100 percent success, you are still much further ahead than most people are.

Having said all that, let's talk seriously about what you need to do to get started on the CAT software project. In general, there are five main steps involved in creating a software application in Visual BASIC, along with a lot of little steps between them.

These five main steps are,

- Define the main functions to be performed by the completed software application.
- Design and then create the user interface.
- Customize the look and behavior of the user interface.
- Place the Visual BASIC code into the application.
- Run and test the software application.

As I outlined in detail in last month's column, nobody writes an application program by sitting down and directly typing into a software program without doing some type of preliminary planning.

First download the background information that Ten-Tec provides for software programmers and study it in detail so you know what tasks you need to perform to control the seven main features found in the RX-320D (refer again to **Table 1**). You then need to sit down with a notebook and write out a plan for controlling those features. Next. create some drawings of what you think the applications user interface (UI) should look like. Set a clear goal for the project, and then set out a mission statement, such as:

"To successfully control the seven main features provided for the operation of Ten-Tec's RX-320D and provide a simple and easy-to-understand user interface when doing so."

You'll want to list everything because later on you'll need to create a checklist to ensure that everything you want to include in your software design is actually in place. If you don't have such a checklist or set of design notes you're simply setting yourself up for frustration and possible failure.

Once you have all of that preliminary information in place you can begin to actually use Visual BASIC to building your CAT program. Now let's look at the design of the UI; next month we'll put BASIC code into that interface to make it work.

Creating The CAT User Interface

One of Visual BASIC's strong points for the hobbyist software programmer is that it provides you with a powerful set of graphic design tools so you can easily design the UI for your software application. This is perfect for software programming in the Microsoft Windows environment, which is, of course, graphically oriented—you don't have to waste valuable time writing software code for those graphical elements (**Photo C**).

Before you actually dive into a programming environment, however, you will need to create a prototype design from which to work. There is a whole art and science to computer interface

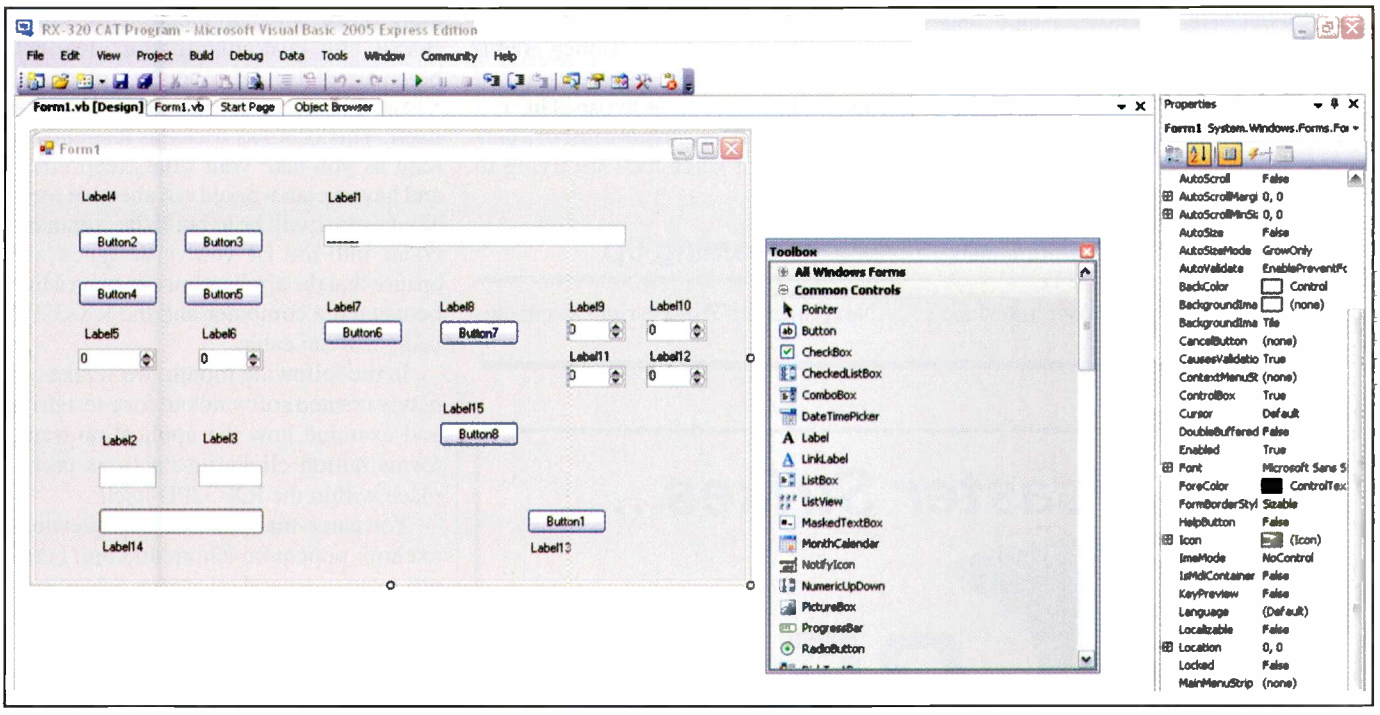


Photo E. The ideas developed through the prototype are put in place in the software application with the Visual BASIC toolbar. You may not be able to make a one-for-one representation in the final application of what you created in the prototype, but try to get as close as possible.

design that's known by a number of different names, such as Human Factors, User Interface Design, and Ergonomics. You need to prototype your design because, very simply, there are as many ways of get something "wrong" as of getting it "right."

An effective interface is visually apparent and forgiving and instills in the user a sense of control. The layout and design of your interface should allow for a natural flow of activity, with an absolute minimum of searching and selecting. In general there should be a consistency to all the functions shown on the screen. For example, buttons you point and click on should perform "on and off" functions and not something unexpected, like control sound volume. Likewise, you want to cluster frequently used buttons, such as frequency tuning, mode selection, and volume control, into one area (**Photo D**).

Take a look at the prototype of my own design (**Photo E**) and see how I've grouped the buttons and display items in a very deliberate way following one of the basic tenets of Human Factors, which is that frequently used buttons should be placed together and in a specific order. Referring to the analysis diagram I've included, you'll can see that physiological studies of the human eye show that it tracks an object in an arch, starting in the lower left hand corner and going to the right. This means that someone using the software program will gather information in this order: Signal Strength, Time, Audio, Bandwidth, Tuning Step, Mode, Tuning Frequency, Audio Volume, Line In Level, BFO, Squelch, Status (Power on/off, Squelch on/off).

Notice that Mute and Tuning are in the "neutral" or middle, area of the user interface. This is because they are not "information," but function, and do not need to be scanned. Likewise, the Frequency display is at the top of the viewing arch, which is the neutral point in the arch and the most restful area to look at. Furthermore, while the primary scanning track is done from left to right, the eye can go either way if there is a need to find specific information. The key point is to place the most important tasks, such as tuning, changing modes, or audio volume, so they are easy to see and access; less important tasks should be placed further away as their importance diminishes.

Once you have laid out your design on paper (or with a computer graphics program), you can begin to "translate" that design into the Visual BASIC working area. As you can see from the illustration of the working area you begin with, you're given a blank "form" to build your UI upon. This form will reflect the default shape and display of the application program when you are finished, so literally "what you see is what you get." As you can see, I've begun to place the buttons and other visual objects that will make up the final UI onto the form. These are drawn using the "toolbox" panel on the left-hand side of the screen, which contains a wide range of items to choose from.

At this point all I am doing is placing selected items onto the form; what I have chosen are "point and click" buttons, text display boxes, numerical display buttons, and text labels. Our task next time will be to customize the look and behavior to customize the look and behavior of the UI and then place the Visual BASIC code into the application.

Take some time to learn more about human factors in computer interface design because it's as important to create good interfaces as it is to write good computer code. Failure to do so can be a waste of your "hidden" design features and extremely frustrating to anyone using your software. There are many sources of information about Human Factors available on the Internet, as well as in many books on the subject.

A good beginning point is the Human Factors and Ergonomics Society (www.hfes.org/web/Default.aspx), a body of professionals who develop "best practices" for many different types of industries. Offering a lighter point of view is the "Bad Human Factors Designs" webpage at www.baddesigns.com/. A good checklist for good human factors design can be found at www.useit.com/ papers/

heuristic/heuristic_list.html, which lists 10 usability heuristics (a fancy word for "rule of thumb") that can help you make the best possible decisions on your design. One of the best sources for software design comes from Microsoft's own on-line library for software programs at http://msdn.microsoft.com/ library/default.asp?url=/library/en-us/ dnanchor/html/anch_uidesigndev.asp.

The key point in any user interface design is the old KISS factor—Keep It Simple Stupid. But remember, radio monitoring is a hobby, so whatever you create, make it enjoyable to use. This is a recreational activity and not a test of a person's endurance when they are trying to tune in stations.

Coming Up

Next time, we'll move right along creating a software application in Visual BASIC by customizing the look and behavior of the UI, and then place the Visual BASIC code into the application itself. This is not as tricky as it sounds as long as you take your time, keep notes, and have a plan worked out ahead of time. The key task will be to build the command codes into the UI you've designed and ensure that the application can open a link between the computer and the RX-320D using a serial cable.

In the following month, we'll take our newly created software out for a test drive and examine how the application transforms button clicks into actions taking place within the RX-320D itself.

You can e-mail me with any questions to carm_popcomm@hotmail.com. I cannot answer general questions on computers, but will be more than happy to help you with any issues raised in the columns.

Even though we're now moving toward summer (and given what took place last year, Heaven knows what we are facing this year), there are still many people who are still feeling the effects of natural disasters such as Hurricane Katrina. I would like to suggest that you continue to send donations to the American Red Cross (www.redcross.org/donate/donate.html) to help your fellow Americans in this time of trouble. There are many other good (and ethical) organizations you can contribute to, so please use them if you wish but don't give into "charity fatigue." If you have a job, a family around you, and are living in a stable neighborhood, then show your thanks for that wonderful good luck by sharing it with someone less fortunate, and do so on a regular basis.

Our troops overseas continue to need our visible support, particularly as we head toward another hot summer. Please refer to the U.S. Department of Defense's official webpage, "Defend America." They have a specific section found at www.defendamerica.mil/support_troops.html which has an amazingly wide range of practical and useful ways that you can directly help.

Again, if you are fortunate to live in the United States of America, have a safe and secure home, a paying job and your loved ones around you when so many don't, please remember to give thanks for your personal blessings by remembering to pass on those blessings to others through regular acts of selfless sharing.

See you again next month! ■

When Disaster Strikes...

REACT is Ready!

REACT Teams work with local, state, and national disaster response agencies. Often ***REACT*** plays a unique role in disaster relief because ***REACT*** is the only volunteer communications organization whose members are trained to use **all types of two-way communications** from CB to packet radio, Amateur radio to GMRS.

Fortunately, disasters don't happen every day. ***REACT*** Teams maintain their readiness and serve the public by monitoring emergency channels and by providing communications services for a variety of activities and community events.

Find out how **you** can be part of the ***REACT*** Team! Visit **www.reactintl.org** to find a Team in your area – or information on starting your own Team.

***REACT* International, Inc.**
(301)316-2900 or (866)REACT99
5210 Auth Rd., Ste. 403 * Suitland, MD 20746